台灣管理學刊
第 10 卷 第 1 期，2010 年 2 月 pp. 19-50

(s,Q)存貨控制缺貨後補模式之多目標最佳化分析[1]

許晉雄
東吳大學財務工程與精算數學系副教授[2]

鄒慶士
國立台北商業技術學院企業管理系教授

摘要

存貨管理對於企業來說是極為重要的工作，其目的是如何運用最少的成本維持高度的服務水準，降低缺貨的可能性以滿足顧客對產品的需求。如何在這些衝突目標間做出權衡取捨，便是多目標存貨控制所面臨的一大挑戰。本研究將Agrell (1995)提出的缺貨後補下三目標(s,Q)存貨控制模式的情況下，運用加入區域搜尋與群集機制的混合式多目標微粒群最佳化來求解不同模式的存貨控制問題，並將結果與傳統存貨控制求解方式及強健柏拉圖進化式演算法比較，發現混合式多目標微粒群最佳化的非凌越解在三項績效衡量指標上明顯的勝過強健柏拉圖進化式演算法，並與傳統存貨控制的求解法不相上下，但傳統的方式一次只能求取一組解，而混合式微粒群演算法基於多點並行的搜尋方式，可以一次求解多組非凌越解並提供多種決策的選擇，同時此演算法要調整的參數較少。

關鍵詞：存貨管理、多目標最佳化、微粒群演算法、缺貨後補模式

壹、緒論

面對市場需求快速變化，企業通常會以存貨來減低需求不確定所導致的缺貨問題，存貨管理的目的便是如何運用最少的成本維持高度的服務水準，降低缺貨的可能性以滿足顧客對產品的需求。一般而言，企業為了維持較高的服務水準，勢必會保有較多的存貨，但往往會造成資金的積壓及管理上的困擾；相反的，企業若基於成本考量保有較少的存貨，容易造成經常性的缺貨，使企業之商譽受損與顧客流失。如何在滿足顧客的需求及降低企業的成本兩衝突之目標間做出權衡取捨，便是存貨控制所面臨的一大挑戰。

在傳統的存貨控制模式中，除了一般的經濟訂購量模式外，最常見的存貨控制機制就是(s,Q)系統，其相關的目標概分為極小化存貨攸關成本並極大化顧客

1本研究承國科會計畫補助，計畫編號：NSC 98-2410-H-141-001。

2聯絡作者:許晉雄，郵寄地址:臺北市 10048 貴陽街一段 56 號，電話:(02) 23111531-3626，傳真:(02) 2381-2510，E-mail: hsiung@scu.edu.tw

服務水準，這些項目彼此之間往往是衝突的，因此在本質上是屬於多目標決策問題，但如何找到禁得起各目標考驗的最佳決策，便是多目標存貨控制中的另一項挑戰。過去在處理多目標的問題時，普遍使用兩種方法。第一種在求解時，先由分析人員在目標函數前設定權重，並將加權後的各個目標予以加總，使多目標規劃問題轉化成單目標的規劃問題，再利用傳統單目標規劃方式求解，稱為權重法(weighted method)；另一種作法是基於計算上的考量，只將其中某項目標最佳化，其餘目標皆轉為限制式，藉由限制其他目標在不同的目標值上進而求得一系列相對應的單一目標最佳解，此稱之為 ε-限制法(ε-constraint method)(Deb, 2004)。這些求解方法最後只能得到近似解，無法保證是非凌越解(non-dominated solution)，而且此種轉換方式下，決策者需要具備某些問題領域知識，才能賦予各目標權重，但在衝突目標的前提下，決策者很難釐清個別目標的相對重要性為何。再者，傳統方式通常一次只能求解一個柏拉圖最佳解(Pareto-optimal solution)，為了取得柏拉圖最佳解前緣(Pareto-optimal front)，必須執行多次最佳化運算，且某些傳統方法對柏拉圖最佳解前緣的形狀很敏感，我們發現要將傳統方法設計成適合多目標最佳化的演算法並不容易，就算找到最佳化的演算法，也無法有效搜尋最佳解。

另外，隨著環境的變遷，存貨控制變得更為複雜，近年來也有許多學者以多目標規劃解決存貨規劃的問題，但大部份的多目標存貨模式是以總合(aggregation methods)的方式求解，如 Padmanabhan 和 Vrat(1990)利用以非線性目標規劃法求解多品項的存貨系統，由於資源的限制，如工作產能、倉儲空間以及資金的限制，所以在存貨規劃時會產生多個衝突的目標，因此使用該法求解需求與存貨水準相關的損耗性存貨管制系統。此法下決策者會設定每一個目標的預期達成水準或目標水準，依此規劃出一個儘可能滿足所有目標水準(即接近目標水準)之結果，並需於事前設定目標之權重與優先次序因子，以求得滿意解。另外，Roy 和 Maiti(1998)提出在模糊儲存空間及總成本預算下，建立需求與存貨水準相關的耗損品多目標存貨模式，並以模糊非線性規劃法(fuzzy non-linear programming, FNLP)及模糊目標規劃法(fuzzy additive goal programming, FAGP)來求解獲利最大化及耗用成本最少化的多目標存貨模式，在模糊空間中總平均成本、倉儲空間、存貨成本、購買價格以及售價格都是不明確的，透過上述的方式求解，並將求解結果與過去學者的研究進行比較，此法必須建立歸屬函數並設定每一個目標之歸屬度，並將各目標結合成單目標，用數學規法計算最佳解。而學者 Mandal, Roy 和 Maiti(2005)則利用幾何規劃求解多品項多目標的模糊存貨模式，在該研究中的其目標為極小化成本且受限於三個模糊限制式，存貨的相關成本設定在某一彈性的範圍中，以求解每個品項的批量及缺貨水準。以上文獻的求解方法必須設定權重、目標值與優先順序，在互動的過程中也須決定目標間的取捨關係，而模糊規劃法則須將各種存貨成本設定在某一範圍中，決策者在欠缺相關資訊的狀況下很難事前設定目標值，目標間相對的重要順序也難以釐清，同時在成本的估計上也具有難度，此外要進行多次繁複的運算才能獲得柏拉圖解，這些都是傳統存貨控制求解方法所遇到的困難點。

在處理(s,Q)多目標存貨控制問題中，往往都希望能極小化存貨成本(包含訂購成本、持有成本與缺貨成本)，過去在處理此類問題時，多是利用單目標最佳化的方式，以求取極小化存貨成本下的最適訂購批量與安全因子，此種方法必須估計有形與無形的缺貨成本，在資訊不完全的狀況下要精確的估算此類無形成本

並不容易，有鑑於此，Agrell(1995)透過缺貨機率來計算缺貨次數與缺貨數量，將極小化缺貨次數與缺貨數量作為存貨控制的另外兩個目標，此種多目標的方法，能讓管理者同時考量存貨攸關成本與缺貨狀況，不需估計惱人的缺貨成本，更可避免管理者因錯估缺貨成本而做出錯誤的情況發生。

但如何求取多目標存貨控制問題的非凌越解又是面臨的另一項難題，近年來利用仿生型的最佳化演算法求解多目標問題成為一種趨勢(Veldhuizen 和 Lamont, 2000)，在過去也已有許多演算法應用在多目標最佳化問題(Zitzler, Laumanns 和 Bleuler, 2004；Shukla 和 Deb, 2007；Ishibuchi, Tsukamoto 和 Nojima, 2008)，而微粒群最佳化(particle swarm optimization, PSO)便屬於仿生型的智慧型演算法，此法下一次可求取多個非凌越解，提供多種決策的選擇，在不同應用領域已被證實是一個表現不錯的演算法(Banks, Vincent 和 Anyakoha, 2007)。而在演算法比較部分，Zitzler 和 Thiele(1999) 曾針對幾個多目標演算法：Strength Pareto Evolutionary Algorithm(SPEA)、Vector Evaluated Genetic Algorithm(VEGA)、Aggregation by Variable Objective Weighting(AVOW)、Niched Pareto Genetic Algorithm(NPGA)及 Non-dominated Sorting Genetic Algorithm(NSGA)之間在六個測試函數找解的優劣，文中發現 SPEA 法在表現上優於其他方法，因此本文挑選此法作為比較基準。另外，在多準則與有限制條件最佳化問題下，微粒群最佳化基於隨機多點並行的演算方式，非傳統單點循序搜尋方式，因此具有強大快速的搜尋能力，可避陷入局部最佳化(local optimum)，應可克服上述傳統求解方式的困難(Banks, Vincent 和 Anyakoha, 2008)。同時此演算法要調整的參數較少，因此本研究將以混合式多目標微粒群最佳化(hybrid multi-objective particle swarm optimization, HMOPSO)的啟發式演算法處理(s,Q)多目標存貨控制問題，可免除傳統求解方式所遭遇到的限制與困難，並可協助管理者作出最佳化的存貨控制決策，提升企業競爭力。本研究目的具體描述如下：1.針對永續盤存制下之(s,Q)系統進行探討，在允許缺貨的狀況下，建構缺貨後補之多目標存貨控制模式。2.以 HMOPSO 求解缺貨後補下三目標存貨控制模型。同時，為驗證 HMOPSO 的效能，將同屬仿生型的強健柏拉圖進化式演算法 SPEA 與 HMOPSO 之非凌越解進行績效評估，並將勝出的演算法與傳統存貨控制方法比較。

貳、研究方法

一、多目標最佳化

真實世界中的最佳化的問題大多具有多目標的概念，需以多個目標函數加以判斷解的優劣，並且這些目標函數之間往往是互相衝突的，這些問題我們稱為多目標最佳化問題。典型的多目標最佳化問題具有 K 個連續的實數值目標函數與 M 個限制式，其數學模式如下(Deb, 2004)：

$$\text{Minimize } \vec{\mathbf{f}}(\vec{\mathbf{x}}) = \left[f_1(\vec{\mathbf{x}}), f_2(\vec{\mathbf{x}}),..., f_K(\vec{\mathbf{x}})\right]^T \quad (1)$$

$$\text{subject to } \vec{\mathbf{x}} \in \Omega \quad (2)$$

$$\Omega = \left\{\vec{\mathbf{x}} \mid g_m(\vec{\mathbf{x}}) \le 0, m = 1,2,...,M\right\} \quad (3)$$

其中 $\vec{\mathbf{x}} = \left[x_1, x_2,..., x_D\right]^T$ 為 D 維度的決策向量，每一個 $x_d (d = 1,2,..., D)$ 可以是實數或整數值，$f_i(\vec{\mathbf{x}})$ $(i = 1,2,..., K)$ 及 $g_m(\vec{\mathbf{x}})$ $(m = 1,2,..., M)$ 是線性或非線性函數。多目標最佳化問題是在求解滿足 M 個限制式的條件下極小化的目標向量

$\vec{\mathbf{f}}(\vec{\mathbf{x}})=\left[f_1(\vec{\mathbf{x}}), f_2(\vec{\mathbf{x}}),..., f_K(\vec{\mathbf{x}})\right]^T$ 的決策向量 $\vec{\mathbf{x}}=\left[x_1, x_2,..., x_D\right]^T$ 。

對於多目標最佳化問題，很顯然地在所有子目標之間會相互有衝突，因此幾乎不可能找到一組設計變數為所有子目標的最佳解，要找的解不是所有目標的最佳解，而是所謂的柏拉圖最佳解，但在真實的多目標問題中，往往無法得知真正的柏拉圖最佳解，所以一般在求解多目標最佳化問題時，主要是找尋目標空間中接近柏拉圖最佳解的非凌越解集合(也有學者使用非劣解或柏拉圖解表示，意思相近)稱為柏拉圖最佳解前緣，因此柏拉圖最佳解前緣中的每一個解又稱為非凌越解，凌越概念的數學表示方式如下：

假設有兩組決策向量 $\vec{\mathbf{u}}=(u_1,u_2,..., u_D)$ 及 $\vec{\mathbf{v}}=(v_1, v_2,..., v_D)$

1. 若 $f_i(\vec{\mathbf{u}})\le f_i(\vec{\mathbf{v}}), \forall i\in\{1,2,...,K\}$ 且 $f_i(\vec{\mathbf{u}})< f_i(\vec{\mathbf{v}}), \exists i\in\{1,2,...,K\}$，則我們稱 $\vec{\mathbf{u}}$ 強凌越(strictly dominate) $\vec{\mathbf{v}}$ ，以符號 $\vec{\mathbf{u}}\prec\vec{\mathbf{v}}$ 表示。
2. 若 $f_i(\vec{\mathbf{u}})\le f_i(\vec{\mathbf{v}}), \forall i\in\{1,2,...,K\}$ ，則我們稱 $\vec{\mathbf{u}}$ 弱凌越(weakly dominate) $\vec{\mathbf{v}}$ ，以符號 $\vec{\mathbf{u}}\preceq\vec{\mathbf{v}}$ 表示。

所以，非凌越解集合是指在該集合中，所有的決策向量均未被集合中其他決策向量所凌越。

二、微粒群演算法

微粒群最佳化演算法與基因演算法都是以隨機多點並行的搜尋方式求解，皆可避免陷入局部最佳化的情況，並由於微粒群最佳化所要調整的參數較少，架構較簡單且容易實現容易實作，在多數的情況下其效能都比基因演算法好(Boeringerh 和 Werner, 2004；Kennedy, Eberhart 和 Shi, 2001)。Sierra 和 Coello(2004)為了避免區域最佳化與增加微粒群最佳化演算法的探索能力，將突變機制加入微粒群演算法中，並與 SPEA2(Strength Pareto Evolutionary Algorithm, SPEA2)、NSGA2(Non-dominated Sorting Genetic Algorithm2)等不同的多目標進化演算法作比較，發現加入了突變機制的微粒群演算法其效果可以媲美 SPEA2 與 NSGA2，甚至可在一些績效指標中發現加入了突變機制的微粒群演算法其效能優於 SPEA2 與 NSGA2。況且，近年來國外有許多研究皆採用微粒群演算法來尋求決策問的最佳解，例如類神經網路設計、天線設計問題、指紋辨識系統、供應商採購決策、轉運問題等，由此可見微粒群最佳化的演算能力是不容忽視的。因此本研究擬採用微粒群演算法來求解多目標存貨控制問題，此種方法只需求調整部份演算法中的參數，並不需要估計缺貨成本、權重、或排定目標的優先順序，一次運算可產生多組非凌越解，應可順利解決傳統求解方法所面臨的困境。

微粒群最佳化演算法概念如下：在一個虛擬的場景中，鳥群於某一區域裡以隨機方式尋找食物，在區域範圍中只有一個地方有食物，而所有的鳥都不曉得食物真正位置，以隨機方式產生不同速度進行問題空間搜尋，一旦有鳥在某一區域發現食物，將驅使更多的鳥在附近覓食，直到整個鳥群都停留在該區塊中，增加了整個鳥群都找到食物的可能性。因此，微粒群最佳化的演算程式就是讓群體成員(在此稱為為粒子)漸漸的往問題空間中較佳的方向移動，以進行求解。在微粒群最佳化演算法中，每隻鳥即為演算法中的一個粒子 x_{nd} ，其中 n 代表第 n 個粒子，d 則代表粒子所搜尋的空間的維度($d=1,2,...,D$)，在求解空間中，進行食物

(最佳解)搜尋。所有的鳥類透過距離食物最近的那隻鳥，可知目前距離食物有多遠。求解的過程中，所有的粒子都會有一個對於最佳化問題的適應值(fitness value)，每個粒子都有一個速度值來決定它們飛翔的方向及距離。粒子不僅根據自己先前的最佳位置(*Pbest*)，並且依據群體經歷的最佳位置(*Gbest*)做修正，每回合粒子可以透過兩種經驗的導引逐漸修正個體的速度與位置，經過一段時間的學習後將會移到全域最佳解的附近。在演算法的參數設定部份，本研究參考 Shi 和 Eberhart (1998b)的慣性權重法(inertia weight method)，而微粒群演算法的流程說明如下：

1. 初始化：以隨機的方式初始族群中每一個粒子在 d 維空間裡的位置與速度。
2. 計算適應函數：針對所設定的目標函數，評估每一個粒子的目標函數值，此函數值亦稱為適應函數值。
3. 尋找個體最佳解(*Pbest*)：各粒子適應函數值與粒子本身的個體最佳解(*Pbest*)記憶比較，更新個體最佳解。
4. 尋找全域最佳解(*Gbest*)：個體最佳解與全域最佳解(*Gbest*)作比較，如個體最佳值優於全域最佳解，則修正全域最佳解。
5. 更新微粒的速度和位置：利用下列公式改變粒子的速度與位置，即

$$v_{nd}^{new} = w \cdot v_{nd}^{old} + c_1 \cdot r_1 \cdot (p_{nd} - x_{nd}) + c_2 \cdot r_2 \cdot (g_d - x_{nd}) \quad (4)$$

$$x_{nd}^{new} = x_{nd}^{old} + v_{nd}^{new} \quad (5)$$

其中 x_{nd} ：第 n 個粒子在 d 維度的位置。

x_{nd}^{old} ：第 n 個粒子在 d 維度原本的位置。

x_{nd}^{new} ：第 n 個粒子在 d 維度新的位置。

v_{nd}^{old} ：第 n 個粒子在 d 維度原本的速度。

v_{nd}^{new} ：粒子 n 在 d 維度新的速度。

w ：慣性權重，介於 0.8~1.2 之間的值。

c_1、c_2 ：學習因子，介於 1~4 之間的值。

P_{nd} ：粒子 n 在 d 維度本身的個體最佳解。

g_d ：第 d 個維度的全域最佳解。

r_1、r_2 ：隨機亂數，介於 0~1 之間的值。

6. 若滿足終止條件，則停止，否則回到步驟 2，繼續執行此演算法，終止條件通常可設為達到最大迭代的次數。

若 $v_{nd}^{new} > v_{\max}$，則 $v_{nd}^{new} = v_{\max}$ ；若 $v_{nd}^{new} < -v_{\max}$，則 $v_{nd}^{new} = -v_{\max}$，其中 $v_{\max}$ 為最大速度，目的為控制粒子的速度在設定的範圍之內，當粒子速度過大時，將可以導正回適當速度向量，而粒子往負向速度過大時，則會限定最大負向速度。在式(4)及式(5)的粒子速度及位置更新方法稱為慣性權重法，是 Shi 和 Eberhart(1998a)提出的更新法則，慣性權重可使粒子保持運動的慣性，使其有擴展搜索空間的趨勢，此法可清除基本 PSO 對 $v_{\max}$ 的需要，因為 w 本身具有維護全域和局部搜索能力平衡的作用，當 $v_{\max}$ 增加時，可通過減少 w 來達到平衡搜索，w 的減少可使得所需的迭代次數變小，因此本研究將 $v_{\max}$ 固定為每維變量的變化範，只對 w 進行調節。在演算法展開初期，通常希望具有較高的探索能力，

因此可將慣性因子(w)的值設定得比較大，隨著迭代次數增加，在演算的後期我們希望粒子能作區域的局部搜尋，並增快收斂速度，可將慣性因子設定的值往下調整來達到目的。此慣性權重法在操作上較為簡單，能避失速的狀況發生，在過去研究中也有不錯的成果，因此本研究採用該法來作為粒子更新速度與位置的方法。我們可以將微粒群演算法的流程以下圖 1 表示：

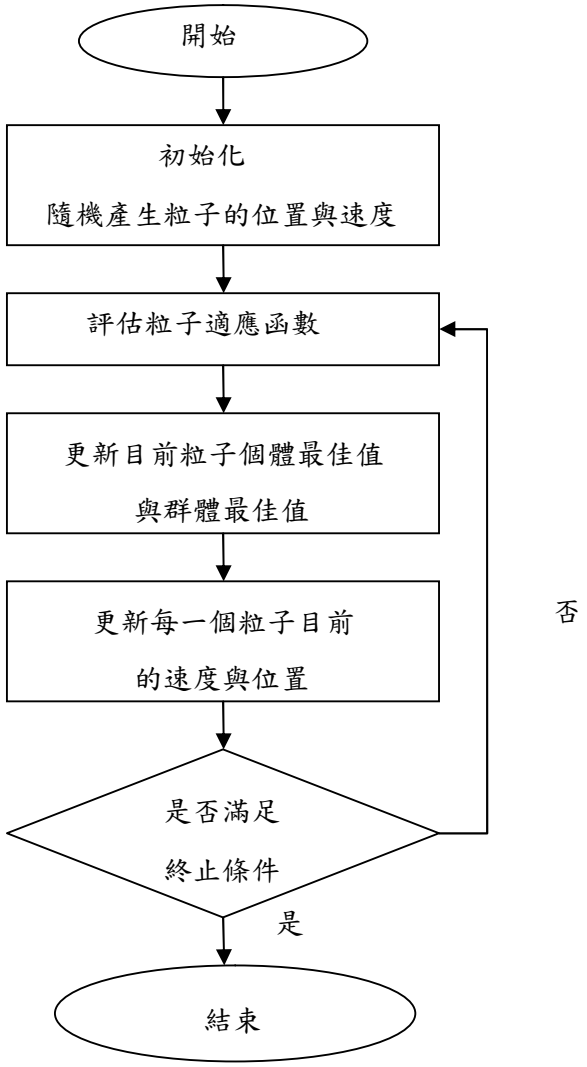

圖 1 微粒群最佳化演算法的演算流程

三、混合式微粒群演算法

微粒群最佳化演算法具有較強的全域搜尋能力，在整個求解空間的搜尋能力相當不錯，但同時亦有陷入局部最佳值的缺點，因此若在微粒群最佳化演算法中加入區域搜尋的機制，必能強化演算法的效能，提升演算法找到最佳解的能力，

加快其收斂速度。此外為了增加解的多樣性與減少電腦運算時間，亦使用加群集機制，以增進演算法效能(Liu, Tan, Goh 和 Ho, 2007)。過去也有許多學者將微粒群演算法加入不同機制與其他演算法比較，發現加入不同機制之微粒群演算法其效能不但優於傳統的微粒群演算法，也和其他進化式的演算法不分上下，甚至有某些效能是超越他們的(Sierra 和 Coello, 2004)，以下先介紹區域搜尋與群集的概念。

(一)區域搜尋(local search)

混合式多目標微粒群演算法運用區域搜尋對非凌越解集合中每一個解的鄰近區域做細微的搜尋以獲得更多及更佳的非凌越解。假設 *LSITER* 是進行區域搜尋的次數，δ是區域搜尋的步距，由於演算法開始時，非凌越解集合尚未穩定，可以在較大的空間搜尋，避免落入區域最佳解，隨這迭代次數增加非凌越解集合逐漸趨近於柏拉圖前緣，此時可縮小搜尋範圍，將步距δ減小，在現有之非凌越解集合附近搜尋，可增加非凌越解的多樣性，使多凌越解更趨近於柏拉圖前緣。

(二)群集(clustering)

當非凌越解數量越多時，判斷某一目標向量是否要加入非凌越解的時間越長，會增加電腦運算時間而降低執行效率，所以在本研究利用群集的機制，能將每次迭代產生的非凌越解縮減至某個固定數量。在 HMOPSO 演算法中，非凌越解集合是引導粒子前進方向的重要指標，所以當非凌越解在某一區段的目標空間密度過於集中時，微粒被導引到該區段的機率將會增加，所求得的非凌越解集合容易陷入局部最佳解，因此使用群集演算法來縮減非凌越解可將距離接近的非凌越解合併為同一族群，只保留位於族群中間的解，讓非凌越解的散佈密度更為平均，增加解的多樣性。此外，整個柏拉圖前緣可能包含大量的解，決策者只需要有適當數量的非凌越解，足以提供決策所需資訊即可，過多的解反而容易造成困擾。Zitzler 和 Thiele(1999)過去採平均聯結法(average linkage method)縮減非凌越解集合有相當不錯的表現，因此本研究也採用該方法，並參考 Tsou 等人(2006)及 Tsou 和 Kao(2008)所提出之方法，結合區域搜尋以及群集機制之混合式微粒群演算法的執行步驟如下：

1. 首先設定迭代次數(T)、粒子數(N)、決策向量的維度(D)(在此為訂購批量及安全因子兩個維度)、儲存非凌越解的陣列$\tilde{A}$。接著隨機產生 N 個粒子的初始位置($\vec{\mathbf{x}}_n^D$)與初始速度($\vec{\mathbf{v}}_n^D$)，此時個體的最佳解$\vec{\mathbf{P}}_n^D$亦為粒子的初始位置$\vec{\mathbf{x}}_n^D$，即$\vec{\mathbf{P}}_n^D = \vec{\mathbf{x}}_n^D$；而為所有粒子比較後所得之全域最佳解的目標向量為$\vec{\mathbf{g}}_n^D = \vec{\mathbf{x}}_n^D$。
2. 設定區域搜尋次數($LSITER$)與搜尋步距(δ)，在非凌越集合($\tilde{A}$)中抽取n個非凌越解進行區域搜尋。判斷微粒($\vec{\mathbf{x}}_n^D$)是否不被所有非凌越解所凌越，若$\vec{\mathbf{x}}_n^D$不被所有非凌越解所凌越，則將$\vec{\mathbf{x}}_n^D$加入$\tilde{A}$中，並移除$\tilde{A}$中被$\vec{\mathbf{x}}_n^D$所凌越的解。
3. 在每一次迭代中，粒子先從$\tilde{A}$中隨機選出一組非凌越解作為全域最佳解($\vec{\mathbf{g}}_n^D$)，再依(4)式、(5)式改變位置($\vec{\mathbf{x}}_n^D$)及速度($\vec{\mathbf{v}}_n^D$)，並依目標函式及式限制式計算出存貨規劃的目標向量($\vec{\mathbf{y}}_n^K$)，再判斷微粒($\vec{\mathbf{x}}_n^D$)是否加入$\tilde{A}$。若$\vec{\mathbf{x}}_n^D$弱

凌越$\vec{\mathbf{P}}_n^D$，或$\vec{\mathbf{x}}_n^D$與$\vec{\mathbf{P}}_n^D$相互不強凌越，則更新粒子本身的個體最佳解$\vec{\mathbf{P}}_n^D$。

4. 最後，以 *ClusterNondominatedSol*()函數縮減$\tilde{A}$至固定數量，以維持非凌越解的多樣性。演算法持續執行到最大迭代數(T)後結束，$\tilde{A}$即為非凌越解集合。

表1　混合式多目標微粒群演算法虛擬碼

```
01  Ã = ∅
02  {x⃗_n^D , v⃗_n^D , p⃗_n^D , g⃗_n^D}_{n=1}^N = Initialize()
03  for t =1 to T
04    δ = ((max_δ - min_δ) * (T - t) / T) + min_δ
05    LocalSearch(LSITER, δ, Ã)
06    for n =1 to N
07      g⃗_n^D = Random(Ã)
08      for d =1 to D
09        v_nd^new = w·v_nd^old + c1·r1·(p_nd − x_nd) + c2·r2·(g_nd − x_nd)
10        x_nd^new = x_nd^old + v_nd^new
11      end for
12      y⃗_n^K = f⃗(x⃗_n^D) = [f1(x⃗_n^D), f2(x⃗_n^D),..., fK(x⃗_n^D)]
13      NondominatedSet(x⃗_n^D)
14      if (x⃗_n^D ≼ p⃗_n^D) or (x⃗_n^D ≺≻ p⃗_n^D)
15        p⃗_n^D = x⃗_n^D
16      end if
17    end for
18    ClusterNondominatedSol()
19  end for
```

參、多目標存貨控制與微粒群最佳化

一、缺貨後補存貨控制模式

Agrell(1995)以互動式多準則決策的方式(Interactive Decision Exploration

Method, IDEM)對(s,Q)存貨控制系統進行研究，該研究中只考慮單一品項，其前置時間內的需求是服從常態分配，在滿足期望年度總成本最小化、期望年度缺貨次數最小化與期望年度缺貨數量最小化三個存貨目標下，同時求解訂購批量及安全庫存量。此法先用單目標的方式求出理想解(ideal solution)，接著尋找一個與理想解最接近的可行解，並提供哪些目標的數值可以被減少以增加其他目標的數值的資訊，直到各目標值均滿意為止。首先介紹 Agrell(1995)所提出的三個目標的存貨控制模式，該研究的假設如下：

1. 模式僅探討單一品項。
2. 該品項在前置時間內之需求(D_L)為常態分配，其平均數為μ_L，變異數為σ_L^2。
3. 前置時間確定已知且是固定的，當存貨水準等於或低於再訂購點(s)時，立即向供應商訂購固定的批量(Q)。
4. 相較於平均存貨，平均缺貨很小，因此可以忽略。

模式中包含三個成本與缺貨的目標分別如下：

目標 1：極小化預期總攸關成本，包含了整年的訂購及持有成本。

目標 2：極小化每年預期缺貨次數，為缺貨機率與年度採購次數($\frac{D}{Q}$)之乘積。

目標 3：極小化每年預期缺貨數量，前置時間內的需求量大於s時，便產生缺貨的情況，缺貨的數量為$(x-s)$。

模式的之決策變數為訂購批量(Q)及安全因子(k)，兩決策變數的上下限限制條件如下：訂購批量必須大於等於零，小於等於預期年需求量；而安全因子必須大於等於零，小於等於預期年需求除以前置時間內需求標準差，此問題之最佳化模式如下：

$$\text{minimize} \quad C(Q,k)=\frac{SD}{Q}+hc\left(\frac{Q}{2}+k\sigma_L\right) \tag{6}$$

$$\text{minimize} \quad N(Q,k)=\frac{D}{Q}(1-\Phi(k)) \tag{7}$$

$$\text{minimize} \quad B(Q,k)=\frac{D}{Q}\int_k^{\infty}(x-s)\,f_{D_L}(x)dx$$

$$=\frac{D\sigma_L}{Q}\left(\varphi(k)-k(1-\Phi(k))\right) \tag{8}$$

$$\text{subject to} \quad 0\le Q\le D\ , \tag{9}$$

$$0\le k\le\frac{D}{\sigma_L}. \tag{10}$$

其中符號代表意義如下：

c：採購單位成本。

S：訂購成本。

h：年度存貨持有成本比率(佔採購單位成本之比率)。

D：預期年需求量。

$\varphi(x)$：標準常態分配之機率密度函數。

$\Phi(x)$：標準常態分配之累積分佈函數。

由於本模式假設前置時間為標準常態分配，可透過安全因子推算前置時間的缺貨機率，進而透過缺貨機率計算缺貨次數與缺貨數量。此種存貨控制模式不必估計缺貨成本，能避免在資訊不完全的狀況下錯估缺貨成本的情形。

二、非凌越解績效評估

本研究主要探討的是在缺貨後補的模式中，評估在不同狀況下，以不同方式求解非凌越解之優劣(包含 HMOPSO、傳統存貨控制求解方式及 SPEA)。參考 Okabe, Jin 和 Sendhoff(2003)及 Deb(2004)等文章，本研究所採用的績效衡量指標包含未被參考集凌越的比率(ratio of non-dominated point by the reference set, $C2_R$)、覆蓋率(set coverage metric)、非凌越解間距(spacing)與最大散佈距離(maximum spread)，以下我們將介紹這四種績效衡量指標。

(一) 未被參考集凌越的比率 $C2_R$：

在比較兩個演算法之績效時，由於傳統存貨控制之求解方法已行之有年，因此我們將傳統的存貨控制求解方法所求得的非凌越解當作比較對象，一般稱為參考集(reference set)，而$C2_R$數值是計算非凌越集合S中的所有非凌越解沒有被參考集集合R所凌越的比率，計算方式表示於(11)式。

$$C2_R(S,R)=\frac{|\{s\in S\mid \nexists r\in R, r\prec s\}|}{|S|} \tag{11}$$

其中第(11)式中分子部分之符號$|\bullet|$指的是在某個模式中非凌越解集合的數量，而未被參考集凌越的比率高，表示非凌越解的品質愈佳，愈接近柏拉圖前緣。

(二) 覆蓋率 $C(U,V)$：

$C(U,V)$ 表示該數值是計算非凌越集合 V 中的所有解被 U 中的解所弱凌越個數的比率，可用來判別兩個集合中解的正確性，以及接近柏拉圖前緣的程度，其計算方式如下：

$$C(U,V)=\frac{|\{b\in V\mid \exists a\in U, a\preceq b\}|}{|V|} \tag{12}$$

由於缺貨次數模式的目標為極小化每年預期攸關成本以及每年預期缺貨次數；缺貨數量模式之目標為極小化每年預期攸關成本以及每年預期缺貨數量，兩模式之攸關成本是相同的目標向量，但缺貨次數與缺貨數量則因為單位不同，因此無法將缺貨次數與缺貨數量的非凌越解之優劣作公平的比較，因此本研究在利用混合式的多目標微粒群最佳化演算法求解出各模式的訂購批量與安全因子後，透過安全因子來計算服務水準，以攸關成本與服務水準來評估非凌越解之優劣，判斷那個模式的非凌越解最接近真實的柏拉圖前緣，其中服務水準(service level；SL)的計算方式列於(13)式(Silver et al., 1998)。

$$SL=1-P(D_L\geq \mu_L+k\sigma_L) \tag{13}$$

(三) 非凌越解間距：

用來評估非凌越解分布密集程度，在計算間距與最大散佈距離前，都必須將各目標函數值予以標準化(normalization)，而後利用(14)式計算非凌越解的間距。

$$S=\sqrt{\frac{1}{|\tilde{A}|}\sum_{i=1}^{|\tilde{A}|}(d_i-\bar{d})^2} \tag{14}$$

其中 d_i 是與鄰近解最短距離，亦即 $d_i=\min_{j\in\tilde{A}\wedge j\neq i}\sum_{k=1}^{K}|f_k^i-f_k^j|$，其中 f_k^i 代表在 $\tilde{A}$ 中第 i 個元素的第 k 的目標，且 $\tilde{A}$ 為柏拉圖最佳化集合中的某一個集合，因為 $\bar{d}$ 是 d_i 的平均距離，因此 $\bar{d}=\sum_{i=1}^{|\tilde{A}|}\frac{d_i}{|\tilde{A}|}$。

(四) 最大散佈距離：

在計算非凌越解集合中相距最遠的兩個非凌越解之間的距離，將標準化後的目標函數利用(15)式計算最大散佈距離，此值愈大表示非凌越解的分佈愈為寬廣。

$$D=\sqrt{\sum_{k=1}^{K}\left(\max_{i=1}^{|\tilde{A}|}f_k^i-\min_{i=1}^{|\tilde{A}|}f_k^i\right)^2} \tag{15}$$

肆、缺貨後補之實例驗證

一、缺貨後補下之三目標存貨控制求解

本節將分別採用 HMOPSO、SPEA 與傳統的存貨控制方法求解在缺貨後補狀況下的存貨控制問題，目標為極小化年度存貨攸關成本、年度缺貨次數、年度缺貨數量。首先呈現 HMOPSO 與 SPEA 的求解結果，接下來介紹傳統的循序法(sequential approach)與同步法(simultaneous approach)求解步驟，並帶入數值加以計算，再做非凌越解的績效評估，並將 HMOPSO 與 SPEA 的求解結果進行比較，最後則將 HMOPSO 與傳統方法求解的結果進行比較。

(一) HMOPSO 求解

表 2 為某財團法人醫院藥劑科之藥品入出庫記錄，本研究利用此組資料進行存貨模式之求解，並比較不同存貨模式差異，進而探討多目標微粒群演算法的實用性，為了方便舉例說明，本文以藥品代號 1 為例進行下列比較與分析。

表 2　藥品範例基本資料表

藥品代號	年需求量 (D)	前置時間內需求量平均數 (μ_L)	前置時間內需求量標準差 (σ_L)	訂購成本 (S)	單位採購成本 (c)	持有成本比率 (h)
1	**3412**	**170.321**	**53.354**	**80**	**27.5**	**0.26**
2	**490**	**24.683**	**5.027**	**80**	**241**	**0.30**
3	**4736**	**231.652**	**57.911**	**135**	**29.41**	**0.30**
4	**200**	**14.853**	**2.969**	**80**	**233**	**0.26**
5	**215**	**12.524**	**2.781**	**80**	**435**	**0.30**
6	**22774**	**1138.706**	**245.333**	**135**	**12.6**	**0.26**
7	**10557**	**341.588**	**85.395**	**135**	**2.14**	**0.26**

HMOPSO 的參數有粒子數、迭代次數、粒子之移動速率、區域搜尋等參數必須設定。在測試參數的過程中，將最大速率分別設定為 100 與 200，整體來說在 100 時，決策變數的變異係數較小；此外，將迭代次數設定為分別設定為 100 與 50，發現在迭代 100 次時，決策變數的變異係數(CV_Q =4.940~5.717，CV_k =3.481~4.278) 小於迭代次數 50 次的（CV_Q =7.407~8.797，CV_k =4.376~5.877)；粒子數則各分別設定為 40 與 60，粒子數為 40 時其決策變數的變異係數(CV_Q =4.940，CV_k =3.481) 小於 60 的變異係數(CV_Q =5.717，CV_k =3.590)，且在粒子數為 40 時執行時間較短。因此本研究將最大速率設定為 100，迭代次數設為 100 次，粒子數設定為 40，移動速率設定為決策變數上限的 1/100，並進行 1 次的區域搜尋，同時為了後續方便比較演算法的績效，我們將最大非凌越解組數設為 30 組。表 3 為以 HMOPSO 求解藥品代號 1 的非凌越解與目標函數值，30 組非凌越解中，訂購批量由 266.15 分佈至 2849.49 不等，而安全因子則是分佈在 0.04 至 2.99 中，存貨攸關成本與年度缺貨次數與數量同時受到訂購批量與安全因子的影響。

表 3　以 HMOPSO 求解之非凌越解及目標函數值

非凌越解代號	Q	k	C	N	B
1	2652.14	2.96	10714	0.0020	0.09
2	341.98	0.47	2200	3.1850	124.57

非凌越解代號	*Q*	*k*	*C*	*N*	*B*
3	452.11	2.09	3017	0.1380	4.11
4	1932.35	2.96	8179	0.0020	0.12
5	344.74	1.64	2650	0.5010	15.19
6	1237.63	2.92	5759	0.0040	0.20
7	341.04	0.95	2382	1.7130	58.98
8	1010.58	2.99	5024	0.0040	0.21
9	331.82	1.09	2425	1.4190	47.36
10	298.72	1.35	2497	1.0130	32.12
11	342.26	0.65	2269	2.5710	95.51
12	319.14	0.26	2095	4.2490	177.57
13	339.16	0.12	2063	4.5500	199.42
14	342.26	0.59	2246	2.7680	104.55
15	336.85	0.04	2030	4.9030	221.13
16	320.98	0.57	2215	3.0230	114.83
17	2849.49	2.97	11416	0.0020	0.08
18	3020.1	3	12032	0.0010	0.07
19	339.42	0.43	2182	3.3540	132.77
20	587.4	2.92	3679	0.0090	0.42
21	827.21	2.9	4394	0.0070	0.32
22	336.18	0.37	2155	3.6100	145.61
23	266.15	1.59	2584	0.7180	21.93

非凌越解代號	Q	k	C	N	B
24	289.22	0.41	2134	4.0220	160.21
25	341.69	0.14	2074	4.4370	193.13
26	336.31	0.78	2311	2.2100	79.31
27	289.61	1.26	2459	1.2250	39.51
28	1654.16	3	7223	0.0030	0.12
29	1472.4	2.95	6575	0.0030	0.16
30	258.38	1.01	2365	2.0650	70.15

附註:訂購批量與安全因子四捨五入至小數第二位,攸關成本四捨五入至整數，年度缺貨次數四捨五入至小數點第四位，缺貨成本均四捨五入至小數點第二位。

(二) SPEA 求解

SPEA 是 Zitzler 及 Thiele 於 1999 年所提出的，它是基因演算法的一種，同樣透過選擇、交配與突變等機制進行演化，特別之處在於其適應值的評估方式，並在演算法中加入群集的機制。SPEA 在演化的過程中，會將非凌越解集合先儲存於外部的族群裡，外部群族中的非凌越解適應值大小是以它凌越基因族群數之多寡而決定的，外部族群的非凌越解凌越內部族群的數量越多，則其適應值越大，基因內部族群的適應值則是以外部族群中所有凌越內部族群的非凌越解之適應值加總，最後合併內部與外部族群進行下一次的演化，並以群集演算法來縮減外部族群中非凌越解的數量，以防止產生過多的非凌越解而降低演算法的執行效能，由於 SPEA 與 HMOPSO 都屬於仿生型的最佳化演算法，且 SPEA 法亦有不錯的搜尋能力(Sierra 和 Coello, 2004)，因此本研究將利用 SPEA 來求解，並與 HMOPSO 比較。SPEA 必須設定交配率與突變率，本研究將交配率設定為 0.7、0.8 及 0.9，突變率則設定為 0.05、0.1 及 0.2，配對出 9 組參數組合，以求解多目標的存貨問題，將 9 組交配率及突變率求解的結果與混合式微粒群演算法比較，發現突變率設定為 0.9，交配率設定為 0.2 時其非凌越解的覆蓋率表現較佳，因此本研究將 SPEA 的交配率與突變率分別設定為 0.9 與 0.2,最大非凌越解組數設定為 30 組，以便與其他方法比較。表 4 為 SPEA 求解藥品代號 1 的非凌越解與目標函數值，在 30 組非凌越解中，其訂購批量由 161.38 分佈至 3095.46 不等，而安全因子則是分佈在 0 至 2.99 中，若兩組非凌越解安全因子相同，則訂購批量愈大，存貨之攸關成本也愈大，年度缺貨次數與數量就愈少；兩組非凌越解訂購批量相同時，安全因子愈大，存貨之攸關成本愈大，年度缺貨次數與數量就愈少。

表 4　以 SPEA 求解之非凌越解及目標函數值

非凌越解代號	*Q*	*k*	*C*	*N*	*B*
1	223.09	2.22	2868	0.2001	6.07
2	261.99	2.75	3028	0.0366	1.43
3	163.78	2.91	3362	0.0345	1.55
4	163.06	2.93	3375	0.0324	1.49
5	163.68	2.95	3378	0.0301	1.41
6	163.79	2.96	3381	0.0290	1.38
7	163.71	2.97	3386	0.0281	1.35
8	163.79	2.99	3393	0.0262	1.28
9	163.78	2.99	3393	0.0262	1.28
10	670.12	2.68	3825	0.0178	0.66
11	161.38	1.53	2852	1.3351	41.1
12	755.35	2.78	4122	0.0115	0.46
13	162.82	1.46	2815	1.5151	47.15
14	550.03	0.72	2737	1.4630	53.33
15	787.87	2.88	4262	0.0079	0.35
16	163.05	1.45	2810	1.5420	48.06
17	163.14	1.44	2806	1.5705	49.03
18	365.46	0.82	2366	1.9253	68.41
19	370.66	0.58	2283	2.5867	97.98
20	1337.59	2.75	6035	0.0072	0.28

非凌越解代號	Q	k	C	N	B
21	1753.13	2.89	7526	0.0035	0.15
22	1968.97	2.86	8269	0.0034	0.15
23	2256.39	2.96	9317	0.0021	0.1
24	3095.46	2.91	12265	0.0018	0.08
25	163.36	0.07	2282	9.8604	439.94
26	162.9	0.05	2277	10.0551	451.86
27	163.78	0.03	2264	10.1671	460.22
28	163.78	0.02	2260	10.2502	465.68
29	163.36	0.01	2259	10.3599	472.38
30	163.62	0	2253	10.4266	477.17

附註:訂購批量與安全因子四捨五入至小數第二位，攸關成本四捨五入至整數，年度缺貨次數四捨五入至小數點第四位，缺貨成本均四捨五入至小數點第二位

(三) 傳統(s,Q)存貨控制方法求解

在進行存貨控制時，存貨的訂購批量與安全因子是兩項重要的決策變數，循序法與同步法便是傳統求取訂購批量與安全因子的方法。在本節中，首先利用服務水準的政策變動，以循序法求解存貨控制問題；接著再透過缺貨成本的估計，以同步的方式求解存貨控制問題。由於這兩種方法分別可透過服務水準政策以及缺貨成本變動的方式求得無限多組解，且這些解之間具有非凌越的特性，因此本研究將這兩種方法求得的非凌越解當作參考集，最後與 HMOPSO 進行比較。

1. 循序法求解：

循序法下，每期存貨的訂購批量是固定的，即為經濟訂購量，計算方式如(16)式所示，而存貨之安全因子則透過利用(17)式計算，管理者應先訂定某一服務水準(P_1)政策，代入(17)式中，即可求得該服務水準下之安全因子。

$$Q=\sqrt{\frac{2SD}{hc}} \tag{16}$$

$$P\{D_L \geq \mu_L + k\sigma_L\}=1-P_1 \tag{17}$$

在此的服務水準(P_1)代表在前置時間不缺貨的機率，前置時間內不缺貨的機率愈高，其服務水準就愈高。

在得到存貨之訂購批量與安全因子後，再將這兩個決策變數代入(6)、(7)與(8)式中，即可求得對應之年度攸關成本、年度缺貨次數與年度缺貨數量之目標函數值。為了避免缺貨造成廠商不可預期的損失，存貨的服務水準政策應維持一定的標準，在 Bookhinder 和 Chen(1992)的文獻中，服務水準政策在 60%～99%間變動，但為了方便後續比較，本研究將服務水準的範圍設定在 51%~99%間，在 51%~89%的範圍中以 2%作為服務水準政策之跳動單位；在 90%~99%的範圍中以 1%作為服務水準政策之跳動單位，如此一來，可產生 30 組解。表 5 是利用循序法，代入藥品代號 1 的參數所得到的結果。循序法下每期的訂購批量都是固定的，即為經濟訂購量，安全因子則透過服務水準計算，當希望維持 55%的服務水準時，安全因子只需設定在 0.13，而當希望維持 80%的服務水準時，安全因子則應設定在 0.84，依此類推。在前置時間平均需求及前置時間需求之標準差已知的狀況下，安全存量(*SS*)利用 $\mu_L + k\sigma_L$ 便可估算。

表5　循序法下之決策變數及目標函數值

非凌越解代號	P_1	Q	k	C	N	B
1	51%	276.32	0.03	1985	6.0505	254.65
2	53%	276.32	0.08	2004	5.8036	238.78
3	55%	276.32	0.13	2024	5.5566	223.51
4	57%	276.32	0.18	2043	5.3097	208.81
5	59%	276.32	0.23	2062	5.0627	194.65
6	61%	276.32	0.28	2082	4.8157	181.01
7	63%	276.32	0.33	2102	4.5688	167.86
8	65%	276.32	0.39	2123	4.3218	155.18
9	67%	276.32	0.44	2144	4.0748	142.95
10	69%	276.32	0.50	2165	3.8279	131.16
11	71%	276.32	0.55	2187	3.5809	119.79
12	73%	276.32	0.61	2209	3.3340	108.83
13	75%	276.32	0.67	2233	3.0870	98.27
14	77%	276.32	0.74	2258	2.8400	88.09

非凌越解代號	P_1	Q	k	C	N	B
15	79%	276.32	0.81	2283	2.5931	78.30
16	81%	276.32	0.88	2311	2.3461	68.89
17	83%	276.32	0.95	2340	2.0992	59.85
18	85%	276.32	1.04	2371	1.8522	51.19
19	87%	276.32	1.13	2405	1.6052	42.90
20	89%	276.32	1.23	2444	1.3583	34.99
21	90%	276.32	1.28	2465	1.2348	31.19
22	91%	276.32	1.34	2487	1.1113	27.49
23	92%	276.32	1.41	2512	0.9878	23.89
24	93%	276.32	1.48	2539	0.8644	20.40
25	94%	276.32	1.55	2569	0.7409	17.02
26	95%	276.32	1.64	2603	0.6174	13.76
27	96%	276.32	1.75	2644	0.4939	10.64
28	97%	276.32	1.88	2693	0.3704	7.65
29	98%	276.32	2.05	2759	0.2470	4.84
30	99%	276.32	2.33	2863	0.1235	2.23

附註:訂購批量四捨五入至整數位，安全因子四捨五入至小數第二位，攸關成本則四捨五入至整數，缺貨次數四捨五入至小數第四位，缺貨數量四捨五入至小數第二位。

由上表可觀察到當廠商欲維持的服務水準愈高，存貨的安全因子愈高，因此安全存量也就愈多，廠商所保有的安全存量愈多，其存貨之攸關成本便會提高，廠商缺貨的次數與缺貨的數量就會隨之減少。一般來說，廠商若想要維持高度的服務水準，勢必要保有較多的存貨以降低缺貨的可能性，此時也會訂購較多的存貨，因此攸關成本就會提高。

2. 同步法求解：

同步法下，需估計每一次缺貨狀況下所產生的成本(B_1)，並利用(18)式與(19)式求解存貨的訂購批量與安全因子，經過多次迭代後，兩決策變數會收斂至某一固定的值，該值就是某一缺貨成本下之訂購批量及安全因子(Silver et al., 1998)。以下介紹同步法的詳細求解步驟：

步驟1：第一次迭代以經濟訂購量作為訂購批量，並估計每發生一次缺貨的狀況會產生多少成本(B_1)，代入(18)式，解出第一次迭代下的安全因子。

$$k = \sqrt{2\ln\left[\frac{DB_1}{\sqrt{2\pi Qc\sigma_L h}}\right]} \tag{18}$$

步驟2：將上次迭代的k值代入(19)式中以求出此次迭代的訂購批量。

$$Q = EOQ\sqrt{\left[1+\frac{B_1}{S}\mathrm{P}\{\mathrm{D_L} \geq \mu_L + k\sigma_L\}\right]} \tag{19}$$

由(19)式求出此次迭代之訂購批量Q後，代入(18)式後便可計算出此次迭代之安全因子。

步驟 3：重覆步驟 2，計算不同迭代所產生的訂購批量與安全因子，直到收斂(最後兩次迭代決策變數值之差小於 0.002)，便停止運算。

步驟 4：將不同的缺貨成本下所對應收斂之訂購批量與安全因子代入(6)、(7)及(8)式中，得到不同的非凌越解。

表 6 為不同缺貨成本下，訂購批量與安全因子收斂的狀況，當缺貨成本估計為 105 元時，訂購批量與安全因子在第 10 次迭代時可收斂，收斂之訂購批量與安全因子分別為 326.07 與 0.53；當缺貨成本估計為 185 元時，訂購批量與安全因子在第 7 次便可收斂，收斂之訂購批量與安全因子分別為 309.20 與 1.23；而當缺貨成本估計為 1055 元，訂購批量與安全因子也在第 6 次就能收斂，收斂之訂購批量與安全因子分別為 297.61 與 2.25。

表 6　同步法下決策變數迭代結果

B_1	決策變數	迭代次數									
		1	2	3	4	5	6	7	8	9	10
105	Q	276.319	313.303	322.336	324.934	325.722	325.965	326.040	326.063	**326.070**	**326.072**
	k	0.780	0.598	0.548	0.534	0.529	0.528	0.527	0.527	**0.527**	**0.527**
185	Q	276.319	304.720	308.578	309.116	309.192	**309.202**	**309.204**	309.204	309.204	309.204
	k	1.320	1.243	1.233	1.232	1.232	**1.232**	**1.232**	1.232	1.232	1.232
1055	Q	276.319	295.923	297.476	297.600	**297.609**	**297.610**	297.610	297.610	297.610	297.610
	k	2.286	2.255	2.253	2.253	**2.253**	**2.253**	2.253	2.253	2.253	2.253

為模擬不同的狀況，表 7 將每次缺貨所產生的成本由 100 元估計至 7555 元，並依步驟 1 至步驟 4 求取不同的缺貨成本下的訂購批量與安全因子，接著將各缺貨成本下之決策變數代入(6)、(7)與(8)式中，得到對應之年度攸關成本、年度缺貨次數與年度缺貨數量之目標函數值。表 7 中可發現，隨著缺貨成本的提高，訂購批量會隨之減少，而安全因子則隨之上升，存貨之攸關成本也會跟著上升，缺貨次數與缺貨數量則隨之減少。以上的結果亦符合管理上的直覺，廠商若認為某一品項的缺貨會造成重大的損失時，勢必會備有較多的存貨以預防缺貨的狀況，由於廠商保有較多的貨品，此時必會造成攸關成本的上升。

針對上述幾種方法，循序法計算較同步法簡便，每次的訂購批量均為經濟訂購量，安全因子則依廠商所想維持的服務水準政策而定，不同的服務水準政策下，所要保持的安全存量便不同，但訂購批量固定為經濟訂購量，並沒有依服務水準而發生變化，而同步法透過估計不同的缺貨成本以計算出不同的安全因子與訂購批量，但此法下必須估計缺貨成本，缺貨成本的估計一般都是管理者主觀的判斷，隨著環境的不斷變化，管理者難以獲取完全的資訊正確判斷缺貨可能產生的成本。由於傳統的存貨控制求解方法有上述的缺點，因此本研究採行混合式的多目標微粒群演算法進行存貨控制的求解，此種方式不需估計缺貨成本，可在極小化各存貨目標下的情況下以隨機的方式同時產生多組訂購批量與安全因子之非凌越解供決策者選擇。

表 7　同步法下之決策變數及目標函數值

非凌越解組數	B_1	Q	k	C	N	B
1	100	331.29	0.39	2155	3.6047	129.43
2	105	326.07	0.53	2204	3.1295	105.84
3	110	322.94	0.62	2238	2.8116	91.33
4	115	320.68	0.70	2266	2.5671	80.81
5	125	317.47	0.82	2309	2.2010	65.99
6	135	315.21	0.92	2344	1.9325	55.81
7	145	313.49	1.00	2373	1.7241	48.29
8	165	310.98	1.13	2421	1.4183	37.86
9	185	309.20	1.23	2458	1.2034	30.95
10	205	307.85	1.32	2489	1.0435	26.04

非凌越解組數	B_1	Q	k	C	N	B
11	255	305.51	1.48	2549	0.7794	18.38
12	305	303.97	1.60	2594	0.6187	14.01
13	355	302.85	1.69	2629	0.5109	11.22
14	455	301.30	1.84	2683	0.3763	7.89
15	555	300.25	1.94	2723	0.2960	6.00
16	655	299.47	2.03	2755	0.2429	4.80
17	855	298.37	2.16	2804	0.1776	3.37
18	1,055	297.61	2.25	2841	0.1391	2.57
19	1,255	297.04	2.33	2869	0.1139	2.06
20	1,655	296.22	2.45	2914	0.0831	1.45
21	2,055	295.65	2.53	2947	0.0650	1.11
22	2,455	295.21	2.60	2973	0.0533	0.89
23	2,955	294.79	2.67	3000	0.0433	0.71
24	3,455	294.45	2.73	3022	0.0364	0.59
25	3,955	294.18	2.78	3041	0.0313	0.50
26	4,555	293.90	2.83	3060	0.0268	0.42
27	5,055	293.71	2.87	3074	0.0239	0.37
28	5,555	293.54	2.90	3087	0.0215	0.33
29	6,055	293.38	2.93	3098	0.0196	0.30
30	7,555	293.01	3.01	3126	0.0153	0.23

附註:訂購批量四捨五入至小數點第二位，安全因子四捨五入至小數第二位，攸關成本四捨五入至整數，缺貨次數四捨五入至小數第四位，缺貨數量四捨五入至小數第二位。

二、HMOPSO 與 SPEA 之比較

圖2是分別以HMOPSO與SPEA求解的30組非凌越解在目標空間中的分佈情形，HMOPSO的解較為集中在圖形的左半邊，SPEA的圖形較為分散，在圖形中間有較大的間斷區域。

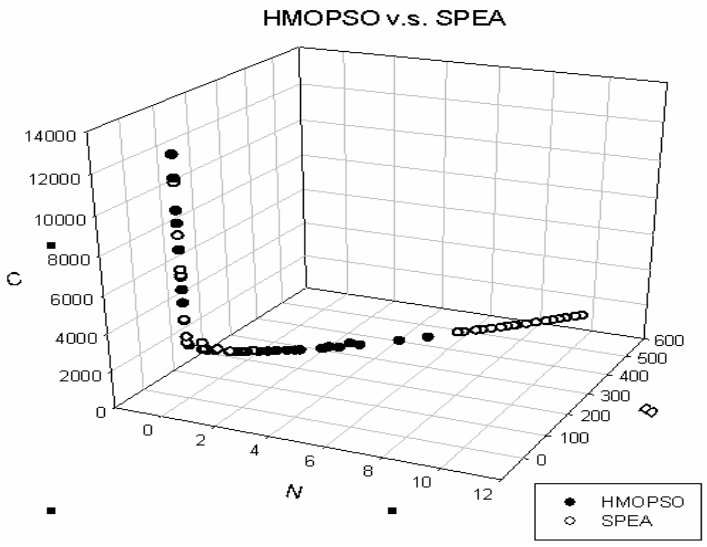

圖2　HMOPSO與SPEA在目標空間的分佈

將 HMOPSO 與 SPEA 所求解的非凌越解透過安全因子(k)計算服務水準(SL)，以攸關成本與服務水準比較二種進化式演算法的求解結果，其服務水準與成本的散佈圖形呈現於圖 3。

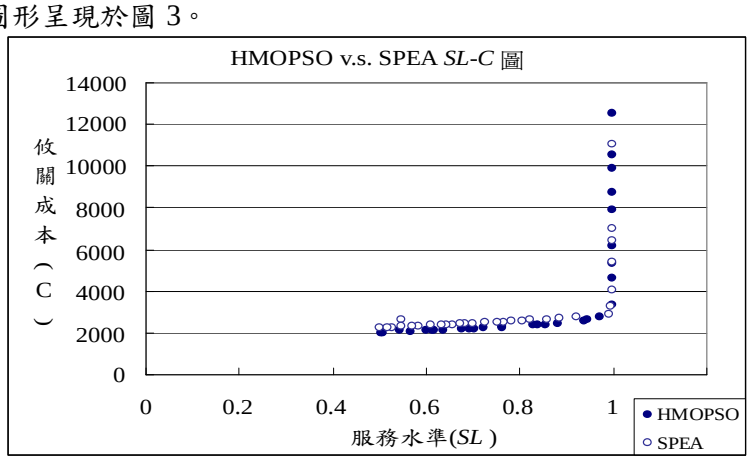

圖3　HMOPSO與SPEA *SL-C*圖

由圖3我們不難發現，服務水準在0.5~0.9的區間中，相同的服務水準下SPEA的攸關的成本較HMOPSO高;此外兩種演算法的結果都顯示出當廠商的服務水準政策維持在0.5~0.9間，存貨之攸關成本是平緩的增加，服務水準愈接近1時，存貨之攸關成本攀升的幅度就愈大，所以當廠商的服務水準政策如果在0.5至0.9之間，提高服務水準其成本增加的幅度並不大，一旦當廠商欲維持高度的服務水準政策，也就是當服務水準愈接近1時，勢必要備妥相當充裕的存貨，持有成本或訂購成本都會迅速的增加。

接著我們進行兩種進化式演算法的非凌越解績效評估，兩種演算法都試行30 次，採用的績效評估指標分別為覆蓋率、非凌越解間距及最大散佈距離。

(一)覆蓋率：

HMOPSO與SPEA分別試行30次後，覆蓋率的結果列於表 8，在試行的30次演算中，HMOPSO的解凌越SPEA的比率平均為89.44%，而SPEA的解凌越HMOPSO的比率平均為2.56%。

表 8　HMOPSO 與 SPEA 覆蓋率

試行次數	*C(HMOPSO,SPEA)*	試行次數	*C(HMOPSO,SPEA)*	試行次數	*C(SPEA,HMOPSO)*	試行次數	*C(SPEA,HMOPSO)*
第 1 次	0.9333	第 16 次	0.9333	第 1 次	0.0333	第 16 次	0.0000
第 2 次	0.8333	第 17 次	0.8667	第 2 次	0.0667	第 17 次	0.0000
第 3 次	0.9000	第 18 次	0.9333	第 3 次	0.0333	第 18 次	0.0000
第 4 次	0.9000	第 19 次	0.8667	第 4 次	0.0667	第 19 次	0.0333
第 5 次	0.8667	第 20 次	1.0000	第 5 次	0.0000	第 20 次	0.0000
第 6 次	0.9667	第 21 次	0.8667	第 6 次	0.0000	第 21 次	0.0333
第 7 次	0.9000	第 22 次	0.9667	第 7 次	0.0667	第 22 次	0.0000
第 8 次	0.8667	第 23 次	0.8667	第 8 次	0.0333	第 23 次	0.0000
第 9 次	0.8000	第 24 次	0.9667	第 9 次	0.0000	第 24 次	0.0000
第 10 次	0.9000	第 25 次	0.8333	第 10 次	0.0333	第 25 次	0.1000
第 11 次	0.9000	第 26 次	0.8667	第 11 次	0.0667	第 26 次	0.0000
第 12 次	0.8333	第 27 次	0.9333	第 12 次	0.0333	第 27 次	0.0667
第 13 次	0.9000	第 28 次	0.8667	第 13 次	0.0000	第 28 次	0.0000
第 14 次	0.9000	第 29 次	0.8667	第 14 次	0.0667	第 29 次	0.0000
第 15 次	0.9333	第 30 次	0.8667	第 15 次	0.0333	第 30 次	0.0000

試行次數	*C(HMOPSO,SPEA)*	試行次數	*C(HMOPSO,SPEA)*	試行次數	*C(SPEA,HMOPSO)*	試行次數	*C(SPEA,HMOPSO)*
平均數	**0.8944*** **			**0.0256**			
標準差	**0.0464**			**0.0299**			

附註：***表 p<0.01，**表 p<0.05，*表 p<0.1

將兩演算法試行30次的覆蓋率作成對樣本的T檢定，t值為79.348，自由度為29，p值為0.000，小於0.01，因此HMOPSO的非凌越解凌越SPEA的非凌解比率顯著的大於SPEA的非凌越解凌越HMOPSO的比率，因此HMOPSO的非凌越解較SPEA的非凌越解更接近柏拉圖前緣。

(二)非凌越解間距：

表9為HMOPSO與SPEA試行30次其非凌越解的間距，HMOPSO解的平均間距為0.0286，HMOPSO的平均間距為0.00348。

表 9　HMOPSO 與 SPEA 非凌越解間距

試行次數	HMOPSO間距	試行次數	HMOPSO間距	試行次數	SPEA間距	試行次數	SPEA間距
第 1 次	0.0067	第 16 次	0.0300	第 1 次	0.0303	第 16 次	0.0448
第 2 次	0.0256	第 17 次	0.0273	第 2 次	0.0385	第 17 次	0.0238
第 3 次	0.0368	第 18 次	0.0302	第 3 次	0.0417	第 18 次	0.0275
第 4 次	0.0261	第 19 次	0.0271	第 4 次	0.0292	第 19 次	0.0311
第 5 次	0.0282	第 20 次	0.0291	第 5 次	0.0309	第 20 次	0.0302
第 6 次	0.0269	第 21 次	0.0347	第 6 次	0.0515	第 21 次	0.0552
第 7 次	0.0371	第 22 次	0.0253	第 7 次	0.0245	第 22 次	0.0256
第 8 次	0.0320	第 23 次	0.0270	第 8 次	0.0300	第 23 次	0.0252
第 9 次	0.0291	第 24 次	0.0254	第 9 次	0.0308	第 24 次	0.0282
第 10 次	0.0331	第 25 次	0.0358	第 10 次	0.0316	第 25 次	0.0367
第 11 次	0.0303	第 26 次	0.0274	第 11 次	0.0487	第 26 次	0.0175
第 12 次	0.0329	第 27 次	0.0319	第 12 次	0.0310	第 27 次	0.0442
第 13 次	0.0274	第 28 次	0.0259	第 13 次	0.0317	第 28 次	0.0296

第 14 次	0.0269	第 29 次	0.0248	第 14 次	0.0446	第 29 次	0.0432
第 15 次	0.0284	第 30 次	0.0272	第 15 次	0.0329	第 30 次	0.0531
平均數	**0.0286 *****			**0.0348**			
標準差	**0.0054**			**0.0096**			

附註：***表 p<0.01，**表 p<0.05，*表 p<0.1

為檢驗其平均數差異是否顯著，因此我們分別做了 HMOPSO 與 SPEA 間距的 T 檢定，t 值為-56.489，自由度為 58，p 等於 0.000，小於 0.01，因此，HMOPSO 非凌越解的間距顯著小於 SPEA。

(三)最大散佈距離：

表 10 為 HMOPSO 與 SPEA 試行 30 次之最大散佈距離，其中 HMOPSO 平均的最大散佈距離為 0.7353，HMOPSO 平均為 0.6181，同樣為檢定平均數是否有顯著差異，我們分別做了 HMOPSO 與 SPEA 最大散佈距離的 T 檢定，t 值為 5.77，自由度為 58，p 值等於 0.000，小於 0.01，因此，HMOPSO 非凌越解的最大散佈距離顯著大於 SPEA。

表 10　HMOPSO 與 SPEA 之最大散佈距離

試行次數	HMOPSO 最大散佈距離	試行次數	HMOPSO 最大散佈距離	試行次數	SPEA 最大散佈距離	試行次數	SPEA 最大散佈距離
第 1 次	0.7423	第 16 次	0.7051	第 1 次	0.6228	第 16 次	0.6539
第 2 次	0.7659	第 17 次	0.7442	第 2 次	0.6098	第 17 次	0.5500
第 3 次	0.7357	第 18 次	0.7465	第 3 次	0.6398	第 18 次	0.6665
第 4 次	0.7572	第 19 次	0.7521	第 4 次	0.5861	第 19 次	0.6992
第 5 次	0.7244	第 20 次	0.7028	第 5 次	0.6782	第 20 次	0.5725
第 6 次	0.7212	第 21 次	0.7271	第 6 次	0.6379	第 21 次	0.6689
第 7 次	0.7646	第 22 次	0.7233	第 7 次	0.7465	第 22 次	0.5760
第 8 次	0.7416	第 23 次	0.7564	第 8 次	0.5503	第 23 次	0.6545
第 9 次	0.7040	第 24 次	0.6998	第 9 次	0.6104	第 24 次	0.7094
第 10 次	0.7241	第 25 次	0.7502	第 10 次	0.6002	第 25 次	0.7092
第 11 次	0.7241	第 26 次	0.7351	第 11 次	0.5898	第 26 次	0.5884
第 12 次	0.7261	第 27 次	0.7134	第 12 次	0.6224	第 27 次	0.5270

第 13 次	0.7673	第 28 次	0.7677	第 13 次	0.5789	第 28 次	0.5611
第 14 次	0.7586	第 29 次	0.6785	第 14 次	0.5988	第 29 次	0.5638
第 15 次	0.7776	第 30 次	0.7235	第 15 次	0.5431	第 30 次	0.6285
平均數	**0.7353*** **			**0.6181**			
標準差	**0.0241**			**0.0557**			

附註：***表 p<0.01，**表 p<0.05，*表 p<0.1

綜合上述結果，加入群集與區域搜尋機制之HMOPSO求解出的非凌越解在覆蓋率、間距以及最大散佈距離的表現都顯著的優於SPEA。因此HMOPSO的非凌越解較SPEA的非凌越解接近柏拉圖前緣，且HMOPSO解的間距較為密集，整體在目標空間的分佈性也較為廣泛，非凌越解的品質優於SPEA。

三、HMOPSO 與傳統循序法及同步法之比較

圖4是HMOPSO求解的非凌越解，以及循序法和同步法的求解結果於目標空間中的分佈情形，HMOPSO解的分佈明顯的比其他兩種方法來得寬廣。

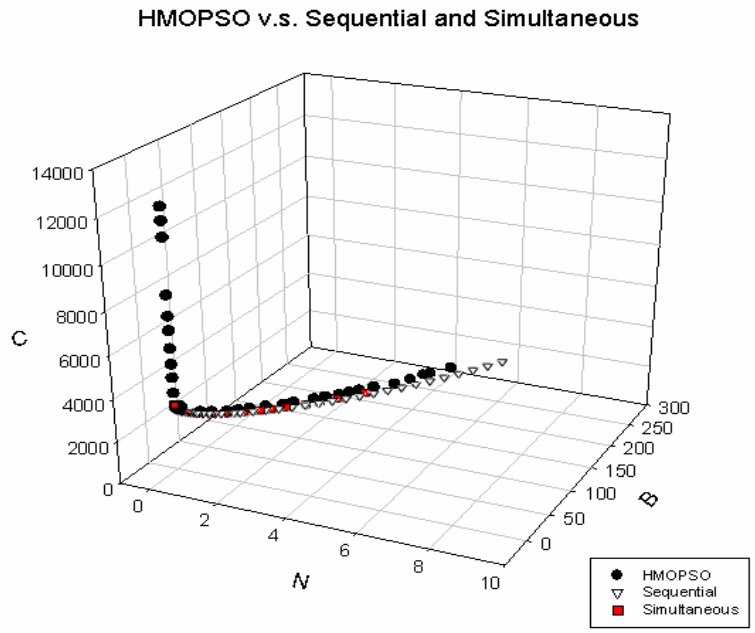

圖4　HMOPSO、循序法與同步法目標空間的分佈

為了更客觀評估 HMOPSO 與循序法及同步法之非凌越解的績效，我們使用三項量化指標進行評估，分別為未被參考集凌越的比率 $C2_R$、非凌越解間距以及

最大散佈距離。

(一)未被參考集凌越的比率：

由於傳統循序法與同步法已廣為使用了很長一段時間，因此本研究在此將循序法與同步法所找到的30組非凌越解當作一個參考集合，分別計算HMOPSO之非凌越解未被循序法及同步法凌越的比率 $C2_R$ *(HMOPSO,Seq)* 與 $C2_R$*(HMOPSO,Sim)*。

1. 以循序法當作參考集之 $C2_R$：

以下是將HMOPSO試行30次，並將這30次的結果與循序法(Seq)比較的結果。表11中我們可以發現混合式的HMOPSO在30次演算中所找到的解不被循序法凌越的平均比率高達0.95，表示HMOPSO所找到的解並不會輕易的被傳統循序法所找到的解所凌越。

表 11 HMOPSO 與循序法之 $C2_R$

試行次數	$C2_R$ (*HMOPSO,Seq*)	試行次數	$C2_R$ (*HMOPSO,Seq*)	試行次數	$C2_R$ (*HMOPSO,Seq*)
第 1 次	0.9333	第 11 次	1.0000	第 21 次	0.9667
第 2 次	0.9333	第 12 次	0.9667	第 22 次	1.0000
第 3 次	0.9667	第 13 次	0.8667	第 23 次	0.9667
第 4 次	0.9667	第 14 次	1.0000	第 24 次	0.9667
第 5 次	1.0000	第 15 次	0.9333	第 25 次	0.9667
第 6 次	0.9333	第 16 次	0.9000	第 26 次	0.9000
第 7 次	0.9000	第 17 次	0.9667	第 27 次	0.9667
第 8 次	0.9667	第 18 次	0.9000	第 28 次	0.9667
第 9 次	0.9333	第 19 次	0.9667	第 29 次	0.9333
第 10 次	1.0000	第 20 次	0.9333	第 30 次	0.9000
平均數	**0.9500**				
標準差	**0.0358**				

2. 以同步法當作參考集之 $C2_R$：

下表是將HMOPSO試行30次，並將這30次的結果與同步法(Sim)比較。同樣的由表12中我們可以發現混合式的HMOPSO在30次演算中所找到的解不被同步法凌越的平均比率高達0.88。

表 12　HMOPSO 與同步法之$C2_R$

試行次數	$C2_R$(*HMOPSO,Sim*)	試行次數	$C2_R$ (*HMOPSO,Sim*)	試行次數	$C2_R$(*HMOPSO,Sim*)
第 1 次	0.8000	第 11 次	0.9333	第 21 次	0.9333
第 2 次	0.9667	第 12 次	0.8667	第 22 次	0.9000
第 3 次	0.8667	第 13 次	0.8667	第 23 次	0.9000
第 4 次	0.9000	第 14 次	0.8667	第 24 次	0.8667
第 5 次	0.9667	第 15 次	0.8000	第 25 次	0.9000
第 6 次	0.8667	第 16 次	0.9000	第 26 次	0.8333
第 7 次	0.9333	第 17 次	0.8000	第 27 次	0.9333
第 8 次	0.9333	第 18 次	0.7667	第 28 次	0.9000
第 9 次	0.9333	第 19 次	0.8333	第 29 次	0.9333
第 10 次	0.9333	第 20 次	0.7333	第 30 次	0.8333
平均數	**0.8800**				
標準差	**0.0591**				

(二)非凌越解間距：

表 9 中 HMOPSO 試行 30 次之間距的平均間距為 0.0286。循序法下非凌越解的間距為 0.0067，同步法下非凌越解的間距則為 0.0274，以非凌越解的間距這項績效指標看來，循序法下的間距(0.0067)小於同步法的間距(0.0274)小於 HMOPSO 的間距(0.0286)。利用 T 檢定來檢驗其平均數是否有顯著差異，因此我們分別做了 HMOPSO 與循序法間距的 T 檢定，以及 HMOPSO 與同步法間距的 T 檢定，HMOPSO 與循序法之 t 值為 4.017，自由度為 29，p 值等於 0.000，小於 0.01，因此，循序法的間距顯著小於 HMOPSO 的間距；而 HMOPSO 與同步法的 t 值為 0.232，自由度為 29，p 值等於 0.818，大於 0.05，所以雖然同步法的間距小於 HMOPSO，但是未達顯著水準。

(三)最大散佈距離：

表 10 中 HMOPSO 試行 30 次的平均最大散佈距離為 0.7353，循序法下非凌越解的最大散佈距離為 0.5079，同步法下非凌越解的最大散佈距離為 0.7310，以非凌越解的最大散佈距離這項績效指標看來，HMOPSO 的最大散佈距離(0.7353)大於同步法(0.7310)大於循序法(0.5079)。檢驗其平均數是否相等，因此我們分別做了 HMOPSO 與循序法最大散佈距離的 T 檢定，以及 HMOPSO 與同步法最大散佈距離的 T 檢定，前者之 t 值為 9.285，自由度為 29，p 值等於 0.000，小於 0.01。因此，HMOPSO 的最大散佈距離顯著大於循序法；後者的 t 值為 0.177，自由度

為 29，p 值 0.86，大於 0.05，所以 HMOPSO 的最大散佈距離雖大於同步法，但是差異未達顯著水準。

綜合上述結果，HMOPSO 求解出的非凌越解不被傳統循序法或同步法所凌越的比率相當高，這兩種傳統的方式過去已廣為使用，表示 HMOPSO 所找的解有一定的可靠程度，也代表著 HMOPSO 可以找到一些傳統方法所找不到的解；非凌越解的間距則以循序法最為密集，同步法與 HMOPSO 次之，其中同步法與 HMOPSO 的差異並不顯著；非凌越解之最大散佈距離則以 HMOPSO 的分佈最為寬廣，同步法及循序法次之，不過 HMOPSO 與同步法的差異並不顯著。

表 13　HMOPSO 與循序法及同步法解之績效評估

排名	績效衡量指標	
	間距	最大散佈距離
1	循序法	HMOPSO
2	同步法	同步法
3	HMOPSO	循序法

雖然以上的結果顯示 HMOPSO 在間距的表現不盡理想，但並不表示 HMOPSO 找到的非凌越解品質不良，參照圖 4，我們可以很明顯地發現 HMOPSO 的非凌越解散佈範圍比循序法及同步法大得多，在這麼大的散佈範圍中，若將 HMOPSO 最大非凌越解組數限定在 30 組，勢必會拉大 HMOPSO 的間距；再者，由於 HMOPSO 加入群集機制，經過多次迭代，演算法會將距離相近的非凌越解合併為同一群族，只保留位於族群中間的解，因此當非凌越解的組數設定較少時，非凌越解的間距會較大。我們嘗試將非凌越解的最大組數調整為 50 組後，HMOPSO 的間距立即縮減至 0.0031。

伍、結論

本研究以進化式演算法來求解(s,Q)存貨控制模式下的非凌越控制參數，在缺貨後補的狀況下，將混合式微粒群演算法求解的結果與 SPEA 比較，此外也與傳統循序法及同步法比較，本研究除了驗證混合式微粒群演算法的有效性外，也提供管理者在選擇存貨控制模式的依據，綜合前述內容所得之結論如下：HMOPSO 在缺貨後補的狀況下與 SPEA 比較，其覆蓋率、間距以及最大散佈距離三項非凌越解績效衡量指標都優於 SPEA。其次，HMOPSO 在缺貨後補的狀況下與傳統的循序法及同步法比較結果，發現它在未被參考集凌越的表現上相當優異，最大散佈距離的表現是 HMOPSO 表現最佳，同步法次之，循序法最差，此外，由圖形也可以看出 HMOPSO 能找到傳統方式所找不到的非凌越解。由於 HMOPSO 加入群集機制，經過多次迭代，演算法會將距離相近的非凌越解合併為同一群族，只保留位於群集中間的解，也因為 HMOPSO 找到的解散佈較為寬廣，因此當非

凌越解的組數設定較小時，其非凌越解的間距就會變大。最後，研究也發現只要將演算法的非凌越組數加以調整後(由 30 組調整至 50 組)，HMOPSO 間距的表現就會比循序法及同步法來得好。

參考文獻

Agrell, P. J. 1995. A multicriteria framework for inventory control**.** ***International Journal of Production Economics***, 41: 59-70.

Banks, A., Vincent, J., & Anyakoha, C. 2007. A review of particle swarm optimization. Part I: background and development. ***Natural Computing,*** 6: 467-484.

Banks, A., Vincent, J., & Anyakoha, C. 2008. A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. ***Natural Computing***, 7: 109–124.

Boeringer, D. W., & Werner, D. H. 2004. Particle swarm optimization versus genetic algorithms for phased array synthesis. ***IEEE Transaction on Antennas and Propagation,*** 52: 771-778.

Bookbinder, J. H., & Chen, V. Y. X. 1992. Multicriteria trade-offs in a warehouse/retailer system. ***The Journal of the Operational Research Society,*** 43(7): 707-720.

Deb, K. 2004. ***Multi-Objective Optimization using Evolutionary Algorithms.*** New York: John Wiley & Sons.

Ishibuchi, H., Tsukamoto, N., & Nojima, Y. 2008. Evolutionary many-objective optimization: A short review. ***Proc. of 2008 IEEE Congress on Evolutionary Computation,*** 2424-2431.

Kennedy, J., Eberhart, R. C., & Shi, Y. 2001. ***Swarm Intelligence.*** San Francisco: Morgan Kaufmann.

Liu, D., Tan, K. C., Goh, C. K., & Ho, W. K. 2007. A multiobjective memetic algorithm based on particle swarm optimization**.** ***IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics,*** 37(1): 42-50.

Mandal, N. K., Roy, T. K., & Maiti, M. 2005. Multi-objective fuzzy inventory model with three constraints: a geometric programming approach. ***Fuzzy Sets and Systems,*** 150(1): 87-106.

Okabe, T., Jin, Y., & Sendhoff, B. 2003. A critical survey of performance indices for multi-objective optimization. ***Proc. of 2003 Congress on Evolutionary Computation,*** 878-885.

Padmanabhan, G., & Prem-Vrat 1990. Analysis of multi-item inventory systems under resource constraints: a non-linear goal programming approach**.** ***Engineering Cost and Production Economics,*** 20: 121-127.

Roy, T. K., & Maiti, M. 1998. Multi-objective inventory models of deteriorating items with some constraints in a fuzzy environment. ***Computers and Operations***

Research. 25(12): 1085-1095.

Shi, Y. H., & Eberhart, R. C. 1998a. A modified particle swarm optimizer. ***IEEE World Congress on Computational Intelligence,*** IEEE Press.

Shi, Y. H., & Eberhart, R. C. 1998b. Parameter selection in particle swarm optimization. ***Evolutionary Programming VII***, 591–600.

Shukla, P. K., & Deb, K. 2007. On finding multiple pareto-optimal solutions using classical and evolutionary generating methods. ***European Journal of Operational Research***, 181: 1630-1652.

Sierra, M. R., & Coello, C. A. 2004. A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms. ***Proceeding of the 2004 Congress on Evolutionary Computation***, 1-39.

Silver, E. A., Pyke, D. F., & Peterson, R. 1998**.** ***Inventory Management and Production Planning and Scheduling***. New York: John Wiley & Sons.

Tsou, C. S., Fang, H. H., Chang, H. H., & Kao, C. H. 2006. An improved particle swarm pareto optimizer with local search and clustering MOPSO-LC. ***The 6***th ***International Conference on Simulated Evolution and Learning, Hefei***, China.

Tsou, C. S., & Kao, C. H. 2008. Multi-objective inventory control using electromagnetism-like meta-heuristic. ***International Journal of Production Research***, 46(14): 3859-3874.

Veldhuizen, D. A. V., & Lamont, G. B. 2000. Multiobjective evolutionary algorithms: analyzing the state-of-the-art. ***Evolutionary Computation***, 8(2): 125-147.

Zitzler, E., & Thiele, L. 1999. Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto Approach. ***IEEE Transactions on Evolutionary Computation***, 3: 257–271.

Zitzler, E., Laumanns, M., & Bleuler, S. 2004. A tutorial on evolutionary multiobjective optimization. In X. Gandibleux, M. Sevaux, K. Sorensen and V. T'kindt (Eds.), ***Metaheuristics for Multiobjective Optimisation***, Springer Heidelberg, 3-37.

Multi-Objective Optimization Analysis of the (s,Q) Backorder Inventory Control Models

Chin-Hsiung Hsu

Associate Professor, Department of Financial Engineering and Actuarial Mathematics, Soochow University

Ching-Shih Tsou

Professor, Department of Business Administration, National Taipei College of Business

Abstract

Inventory management is an important work to the enterprise. Traditional inventory models only involve single objective which relates to several cost concepts and/or service requirements. Even in its multi-objective formulation, most models have been solved by aggregation methods. Such solutions obtained are unsatisfactory because decision makers try to act through a surrogate variable with incomplete information. So inventory management could be regard as a multi-objective optimization (MOP) problem. This work analyzes Agrell's inventory control problem and applies hybrid Multi-Objective Particle Swarm Optimization (HMOPSO), which incorporates a local search and clustering method, to an inventory planning problem. The way of multi-objective analysis can determine lot size and safety factor simultaneously under the objectives of minimizing the expected total relevant cost and some measurements about stockout. HMOPSO algorithm is compared with the traditional inventory control approach (such as the simultaneous and sequential approach) and Strength Pareto Evolutionary Algorithm (SPEA). The comparative results show that the HMOPSO surpasses the SPEA on the three performance indexes, and is competitive with than traditional approaches. However, HMOPSO can find lots of non-dominated solution in a single run and traditional approaches just search for one in a single run.

Keywords: Inventory Management, Multi-Objective Optimization, Particle Swarm Optimization, Backorder Model